Flower Classification

Franko Ndou

BlossomTech

- Floral Focused AI Tech Startup
- Creates ML models to support its other endeavors (App development and E-Commerces)
- Core mission to pursue innovation to further understand the beauty of nature

BlossomTech

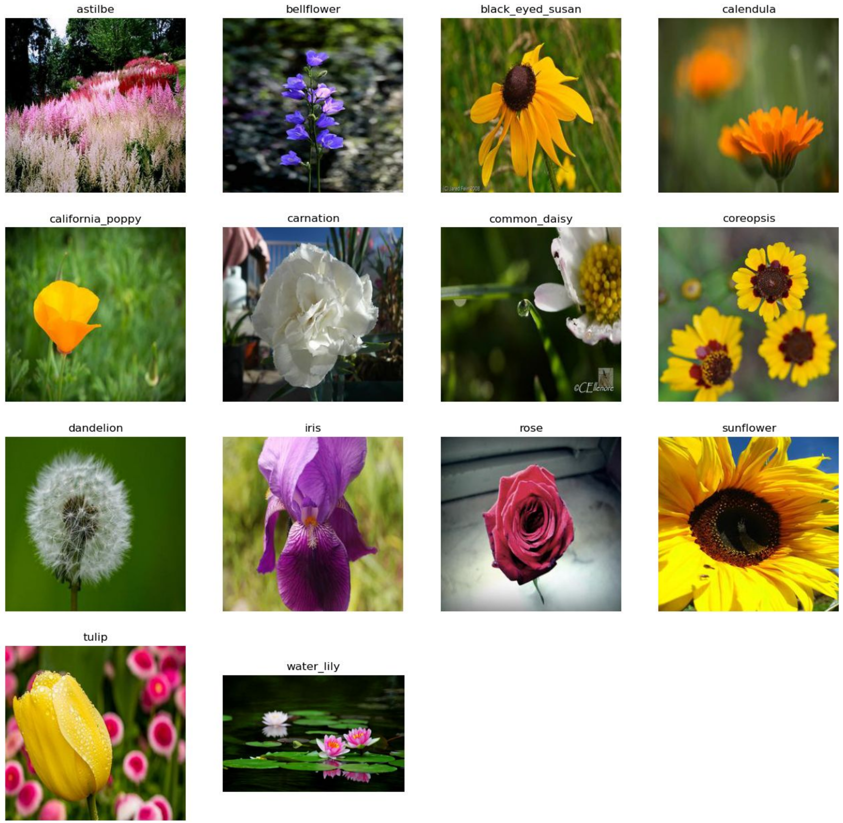
Business Problem

- BlossomTech: Tech startup based in NYC
- The target sectors include agriculture, horticulture, and e-commerce.
- Objective: Develop an AI model to supersede current slow and error-prone identification methods

BlossomTech

Dataset

- Utilizing the "Flower Classification" dataset from Kaggle.
- Features diverse flower images across 14 species, each labeled for supervised learning.
- Dataset includes training and validation (used as test data) sets.
- Total of approximately 13,000 images, with 207 images in each test folder and 500-800 images in each training folder.



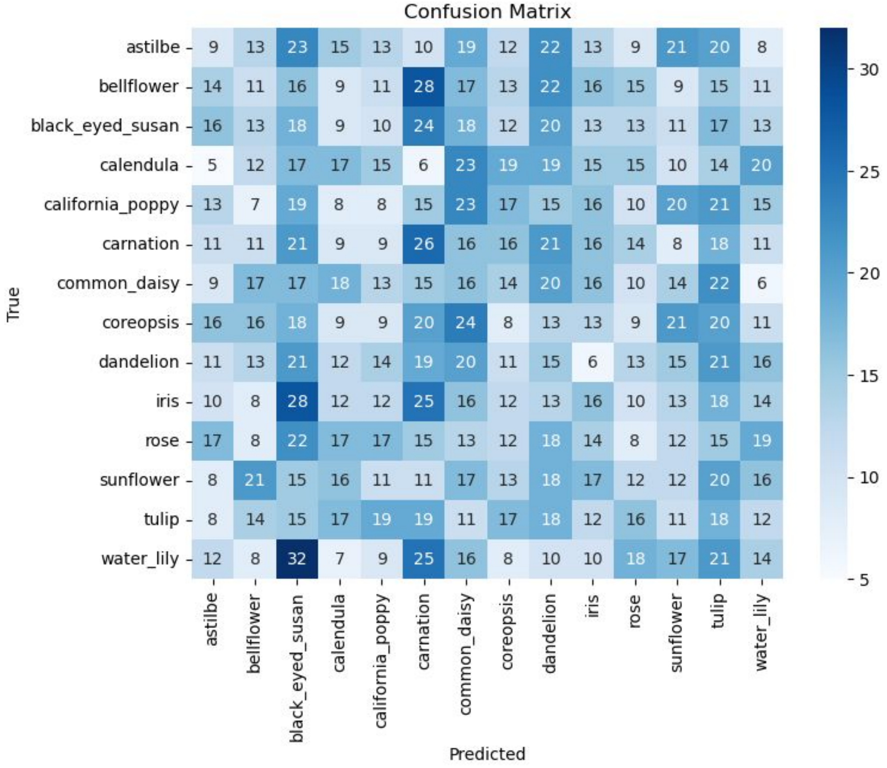
Architecture

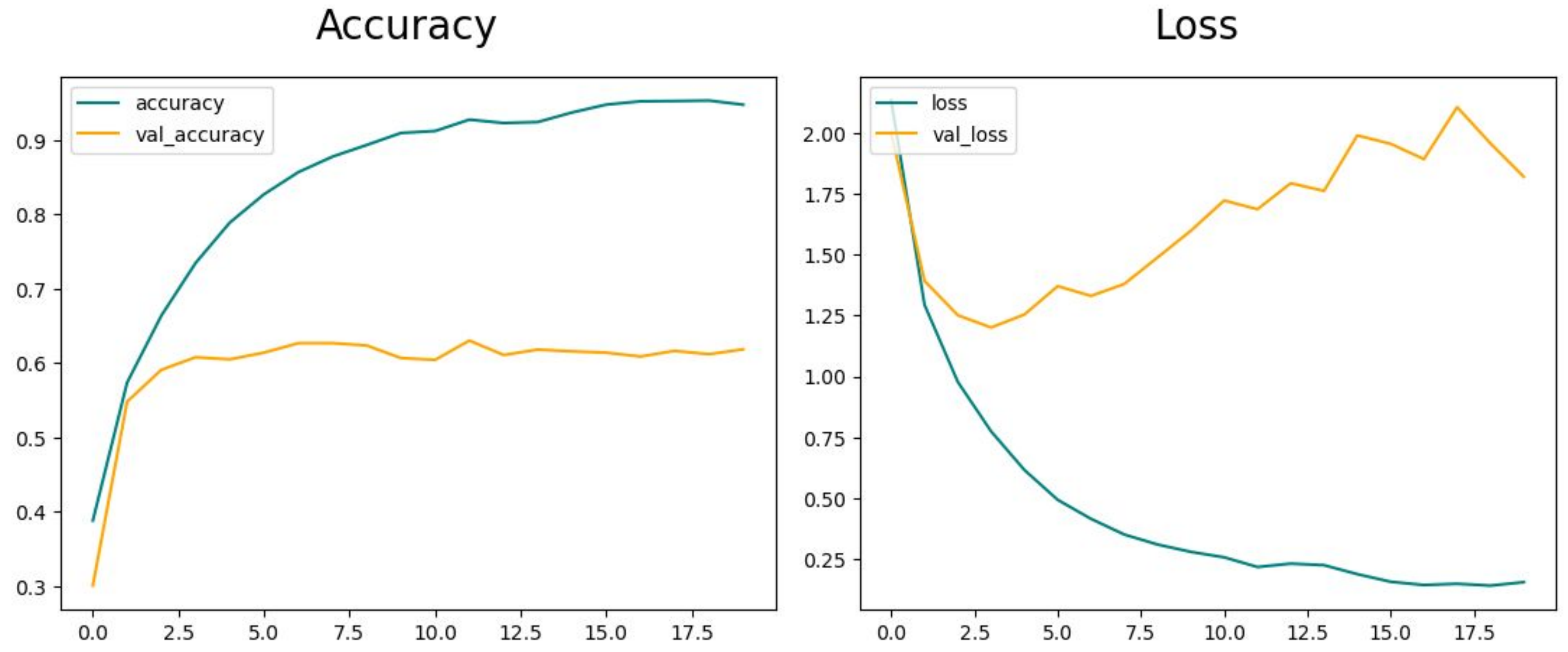
- Sequential neural network with 3 convolutional layers, pooling, batch normalization, 1 dense hidden layer, and an output layer.
- Utilizes ReLU and Softmax activations, dropout regularization.
- Configured for classifying images into 14 categories.

```python
# Function to create a Keras model
def create_model(optimizer='adam', dropout_rate=0.5):
    model = Sequential()
    model.add(Conv2D(16, (3, 3), 1, activation='relu', input_shape=(128, 128, 3)))
    model.add(MaxPooling2D())
    model.add(BatchNormalization())
    model.add(Conv2D(32, (3, 3), 1, activation='relu'))
    model.add(MaxPooling2D())
    model.add(BatchNormalization())
    model.add(Conv2D(16, (3, 3), 1, activation='relu'))
    model.add(MaxPooling2D())
    model.add(BatchNormalization())
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(14, activation='softmax'))

    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

    return model
```

Baseline Model (Pre-Augmentation)

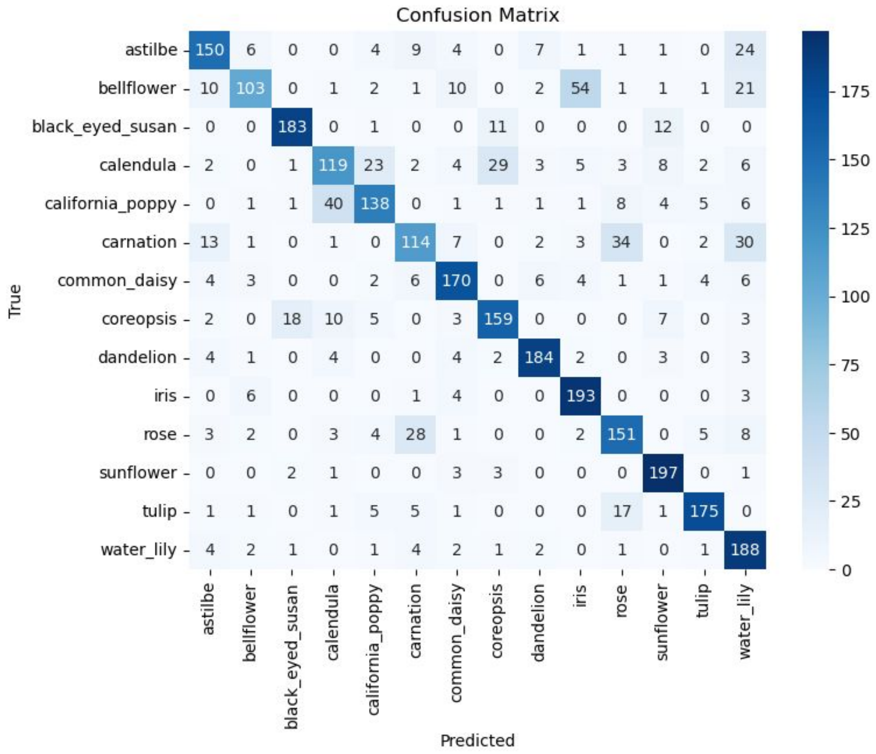
- Started with poor baseline model performance: only 59% accuracy.
- Faced significant overfitting issues.
- Accuracy and loss scores indicated severe overfitting.




Accuracy
accuracy
val_accuracy
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Loss
loss
val_loss
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

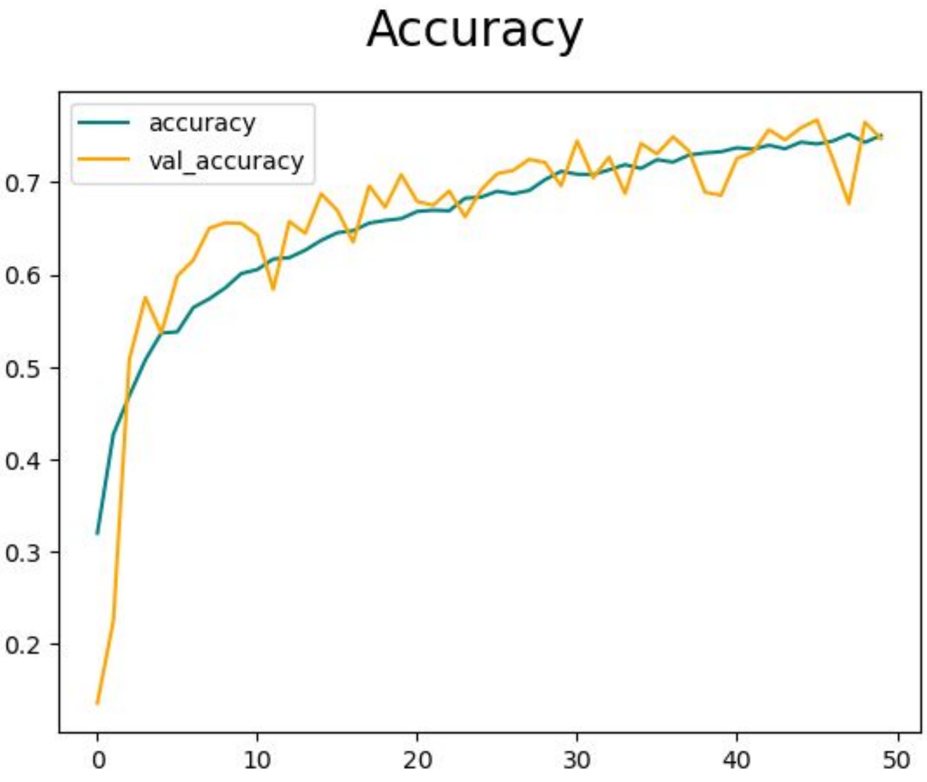
Baseline Model (Post-Augmentation)

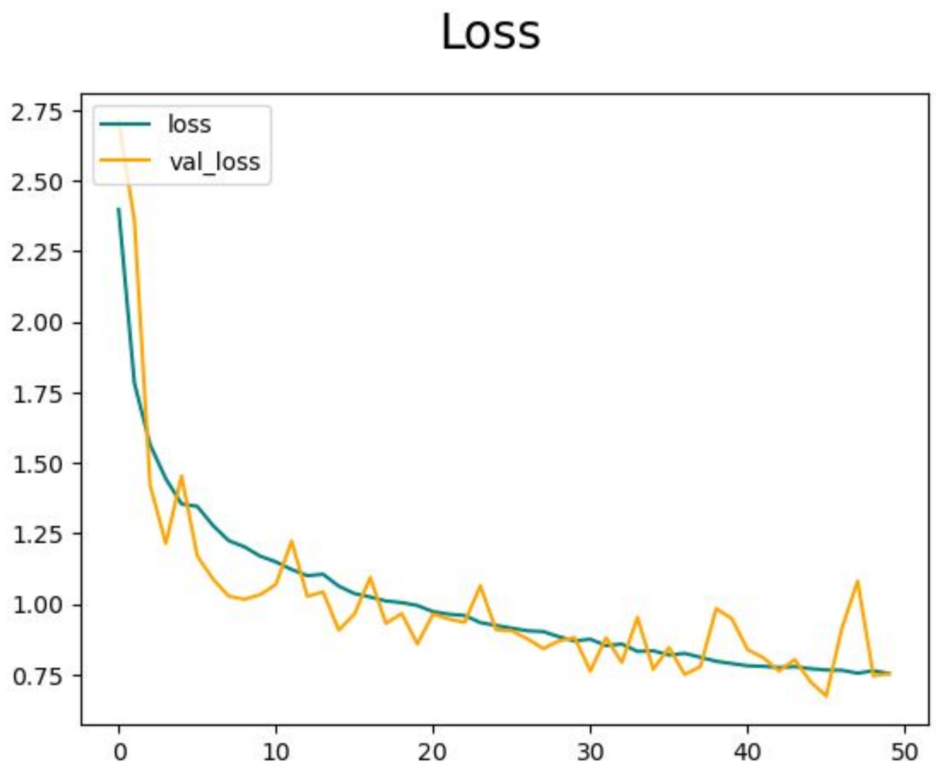
- Model accuracy increased significantly from 59% to 76.7%.
- Notable reduction in overfitting.
- Enhanced generalization of data patterns by the model.



Calendula

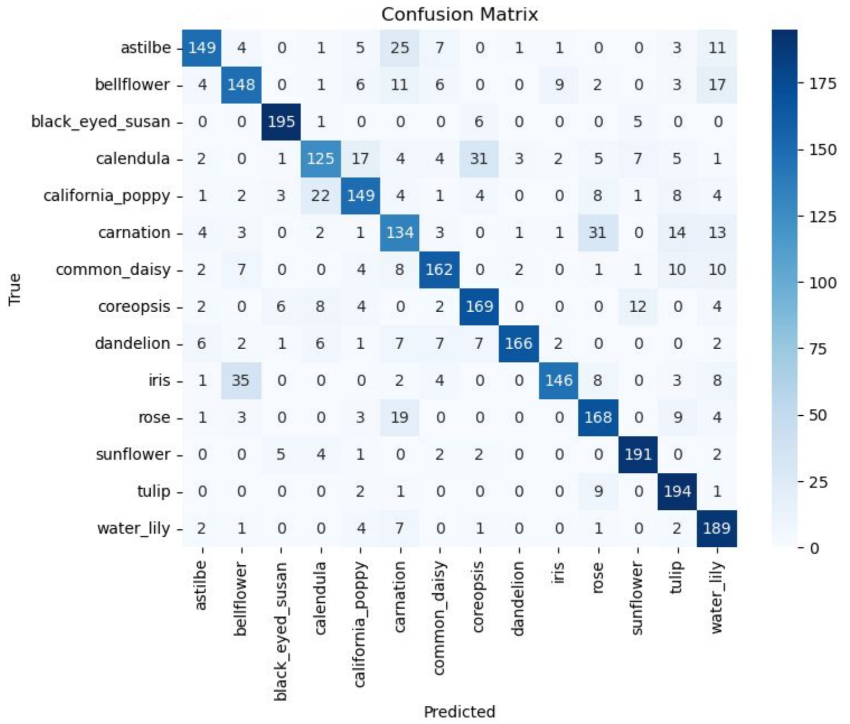
California Poppy


Accuracy
accuracy
val_accuracy
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50


Loss
loss
val_loss
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0
10
20
30
40
50

Hyperparameter Tuning

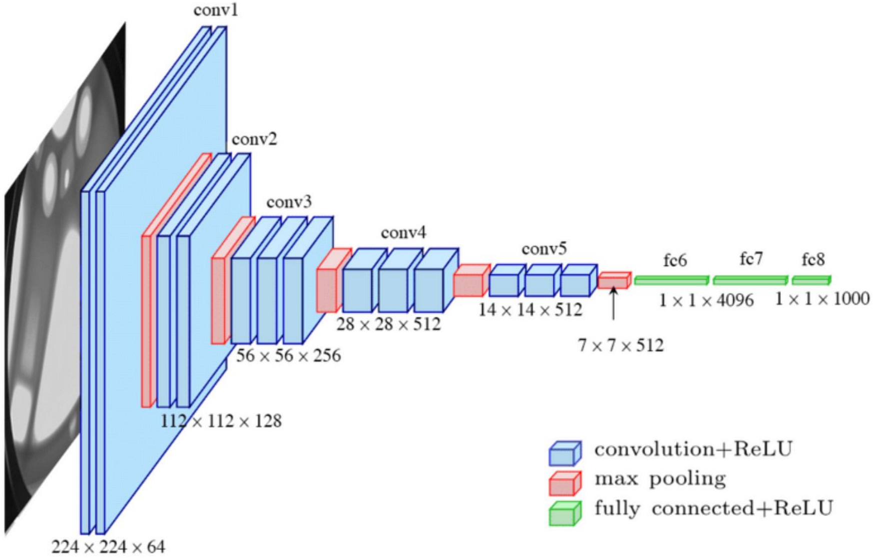
- Improved model accuracy from 76.7% to 78%, a 1.3% increase.
- Integrated callbacks including EarlyStopping and ReduceLROnPlateau.
- Fine-tuned key hyperparameters: units, dropout rates, and regularization rates.

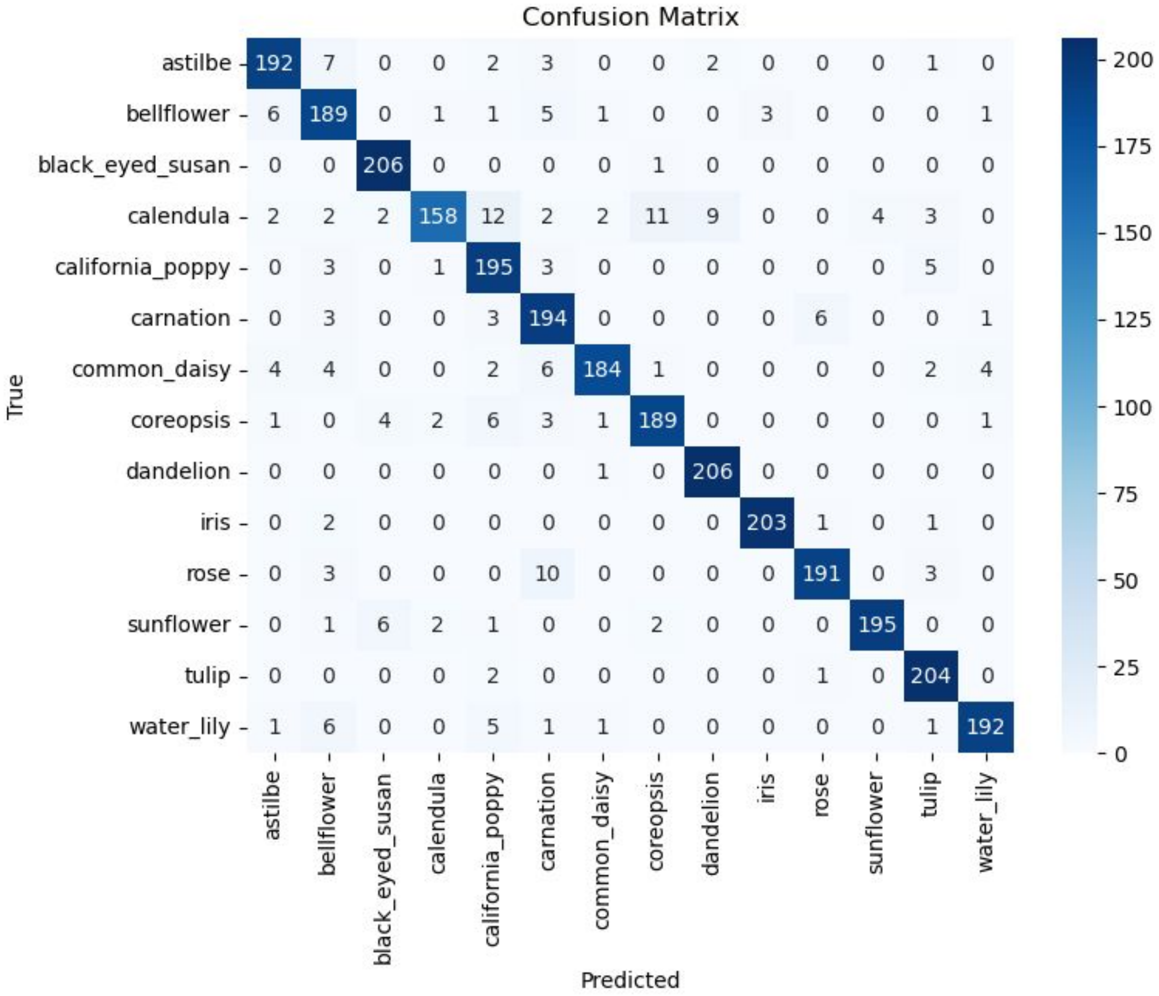


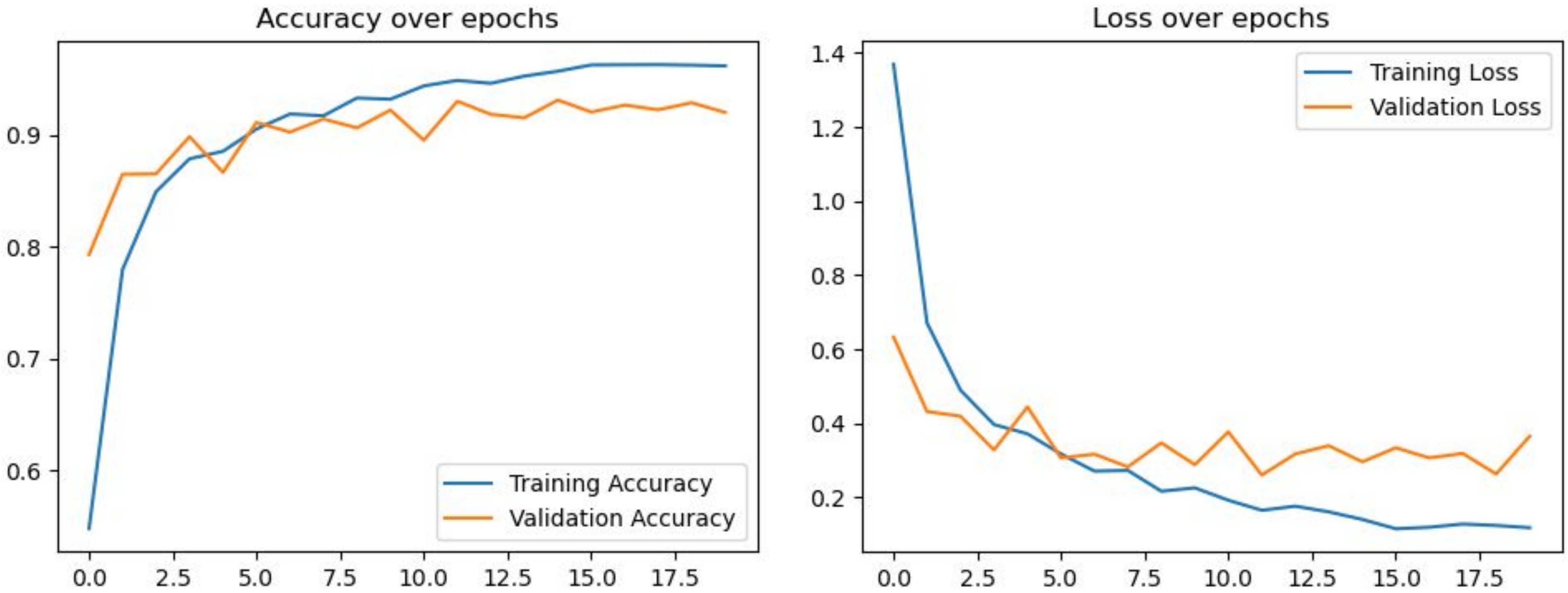

Accuracy over epochs
Training Accuracy
Validation Accuracy
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
10
20
30
40
Loss over epochs
Training Loss
Validation Loss
1
2
3
4
5
0
10
20
30
40

Best Model (VGG16)

- VGG16 is a deep convolutional neural network with 16 weighted layers often used as a benchmark in computer vision.
- Trained from imagenet dataset (14 million images)
- Performed best out of all models when incorporated into our dataset (93.1% accuracy)




Confusion Matrix
True
Predicted
astilbe
bellflower
black_eyed_susan
calendula
california_poppy
carnation
common_daisy
coreopsis
dandelion
iris
rose
sunflower
tulip
water_lily
192 7 0 0 2 3 0 0 2 0 0 0 1 0
6 189 0 1 1 5 1 0 0 3 0 0 0 1
0 0 206 0 0 0 0 1 0 0 0 0 0 0
2 2 2 158 12 2 2 11 9 0 0 4 3 0
0 3 0 1 195 3 0 0 0 0 0 0 5 0
0 3 0 0 3 194 0 0 0 0 6 0 0 1
4 4 0 0 2 6 184 1 0 0 0 0 2 4
1 0 4 2 6 3 1 189 0 0 0 0 0 1
0 0 0 0 0 0 1 0 206 0 0 0 0 0
0 2 0 0 0 0 0 0 0 203 1 0 1 0
0 3 0 0 0 10 0 0 0 0 191 0 3 0
0 1 6 2 1 0 0 2 0 0 0 195 0 0
0 0 0 0 2 0 0 0 0 0 1 0 204 0
1 6 0 0 5 1 1 0 0 0 0 0 1 192
200
175
150
125
100
75
50
25
0


Accuracy over epochs
Training Accuracy
Validation Accuracy
0.6
0.7
0.8
0.9
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Loss over epochs
Training Loss
Validation Loss
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Recommendations:

- Integrate flower recognition in BlossomTech's app for instant flower info, care instructions, and purchase options, boosting customer engagement and sales.

- Implement the technology in BlossomTech’s gardening apps for immediate flower identification and care tips, enhancing user experience and customer base.

- Use user interaction data from the flower recognition feature to customize marketing, product recommendations, and shopping experiences on BlossomShop.

- License the flower recognition technology to florists, landscaping firms, and botanical gardens for additional revenue.

Next Steps:

Architecture Modifications:

- Increase Model Complexity: Experiment with adding more convolutional layers, increasing the number of filters, or using deeper neural network architectures. Be cautious not to overfit, and use dropout layers to regularize if needed.

- Adjust Pooling Layers: Try different pooling techniques like AveragePooling2D or GlobalAveragePooling2D instead of MaxPooling2D to capture different features.

Ensemble Learning:

- Combine Multiple Models: Train multiple models with different architectures or hyperparameters and ensemble their predictions. This can help improve overall accuracy

Custom Loss Functions:

- Design custom loss functions that emphasize specific aspects of your task, such as reducing false positives or false negatives, if your dataset has imbalanced classes.

Contact Me:

Franko Ndou

GitHub: https://github.com/fndou05

LinkedIn: https://www.linkedin.com/in/frankondou/

Email: frankondou@gmail.com